Algoverse Proposal

Jarod Umland[1,2], Julius Brehme[1,3], Matt Ruiz Dias[1,4], and Shin Yun Seong[1]

[1] Sungkyunkwan University, 25-2 Seonggyungwan-ro, Jongno-gu, Seoul, South-Korea
[2] Hochschule für Wirtschaft und Recht, Badensche Str. 52, 10825 Berlin, Germany
[3] Julius University
[4] University of the Basque Country, Donostia-San Sebastian GI 20018, Spain
j.umland@yahoo.com, ruizdiasmatt@gmail.com

Abstract. The following paper introduces the proposal of a capstone project of a learning platform for visualizing algorithms named Algoverse, that is going to be developed at the Sungkyunkwan University in the fall semester of year 2023 during the course Capstone Design Project SWE3028-41.

Keywords: Algorithms · Learning Platform · Web Application

1 Introduction

In the ever-evolving landscape of the 21st century, computer science has emerged as a pivotal discipline, and algorithms stand as a cornerstone of this field. Algorithms are indispensable tools in problem-solving and program optimization, with their selection significantly impacting the efficiency of software. However, the abstract nature of algorithmic concepts often poses challenges for students. Traditionally, learners rely on classroom instruction and textbooks for algorithmic comprehension. However, the need for more intuitive, visually oriented educational resources is evident. This project proposal outlines the creation of an Algorithm Visualization Platform designed to facilitate algorithmic learning by providing students with clear, visual, and intuitive representations of these fundamental concepts.

2 Motivation

2.1 Problematic learning system

As briefly mentioned before, professors still rely on highly theoretical teaching methods such as classroom instruction and textbooks. These outdated methods bring some issues on their own. Issues that can make the task of learning these new concepts an even more complicated task.

One of the main issues is that it can be challenging for students to understand the material. Theoretical concepts are often abstract and difficult to visualize,

which can make it hard for students to grasp the underlying principles. This can lead to confusion and frustration, which can ultimately result in lower grades and a lack of interest in the subject.

Additionally, highly theoretical teaching methods can be outdated and ineffective. They may not take into account the latest research and developments in the field, which can lead to a lack of relevance and practical application. This can be especially problematic in fields such as computer science, where new technologies and techniques are constantly emerging.

Finally, highly theoretical teaching methods can be discouraging for students who are interested in pursuing careers in the field. If students are unable to understand the material or see its practical application, they may become disenchanted with the subject and choose to pursue other career paths.

In summary, highly theoretical teaching methods can be problematic for a number of reasons. They can make it difficult for students to understand the material, be outdated and ineffective, and discourage students from pursuing careers in the field.

2.2 Importance of learning platforms

Learning platforms, especially those that are based on algorithm and data structure visualization, are essential tools for educational institutions and businesses alike. These platforms offer a wide range of benefits that can help streamline the learning process and make it more efficient.

One of the most significant advantages of these platforms is that they allow learners to visualize complex concepts in a more intuitive way. By using visual aids such as diagrams, animations, and interactive simulations, learners can better understand how algorithms and data structures work. This can help them to grasp these concepts more quickly and easily than they would with traditional teaching methods.

Moreover, these platforms can help learners to develop practical skills that are essential in today's job market. Many employers require candidates to have experience with algorithms and data structures, especially in fields such as software engineering, data science, and artificial intelligence. By using these platforms, learners can gain hands-on experience with these concepts and develop the skills they need to succeed in their careers.

In conclusion, learning platforms based on algorithms and data structure visualization are important because they offer a more intuitive way for learners to understand complex concepts. On top of that, students can develop practical skills that are essential in today's job market.

2.3 Importance of algorithms and data structures

Algorithms and data structures are fundamental concepts in computer science. They are essential for developing efficient and optimized computer programs. Learning these concepts can help you become a better programmer and prepare you for a career in fields such as software engineering, data science, and artificial intelligence.

Additionally, according to a survey that we conducted among Computer Science related students and new graduates; more than 90% agree that these concepts are important for their career path. Part of this survey with this precise information can be seen below in Figure 1.

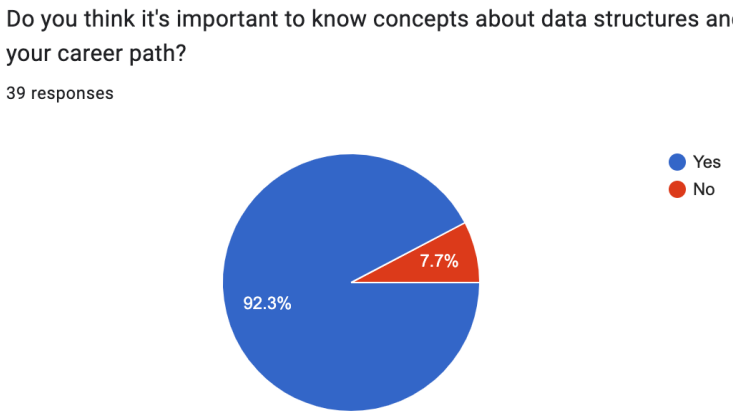


Fig. 1: Question regarding the importance of algorithms and data structures asked in the survey.

2.4 Survey Analysis

In this section, we will go over the survey previously mentioned when stating how important algorithms and data structure concepts are. Besides, in order to avoid redundancy, the question already shown in Figure 1 will be skipped.

Before diving deep into the survey and its questions, we will briefly introduce what was the purpose of it to begin with. Simply put, the survey was made in order to prove the points we were trying to make. Among them, are the issues the students face when learning algorithms and data structure concepts, the outdated learning system, the importance of these concepts for a hypothetical future career, etc.

1) Tools used when trying to learn a new concept

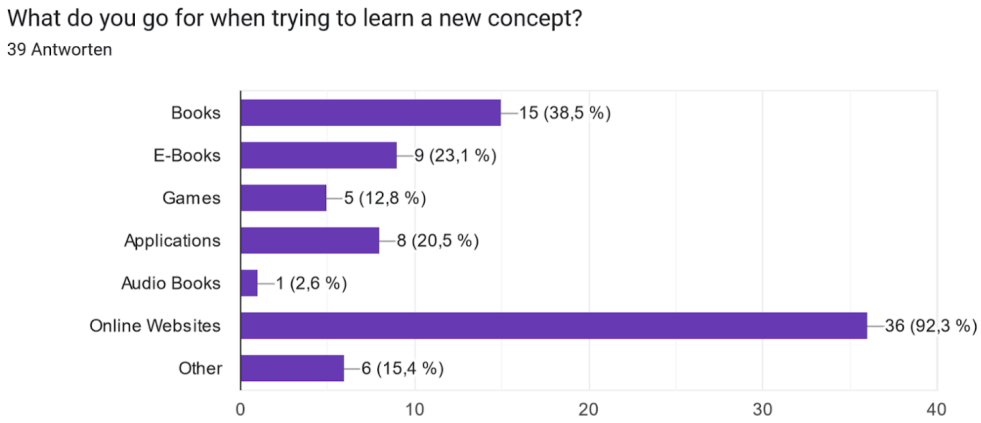


Fig. 2: Question regarding the tools preferred when trying to learn a new concept.

2) Difficulty of algorithm and data structure concepts

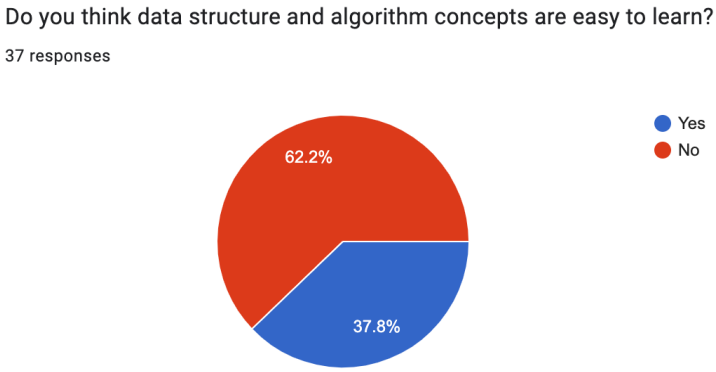


Fig. 3: Question regarding the difficulty of data structure and algorithm concepts.

3) Easier to understand visualized topics

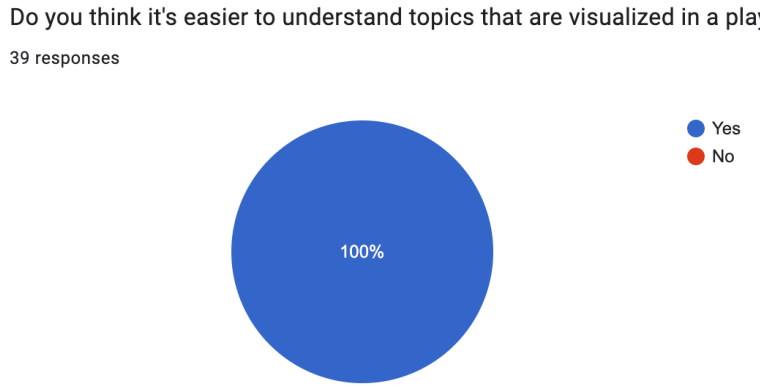


Fig. 4: Question regarding the understandability of visualized topics.

4) Use of visualization apps or websites

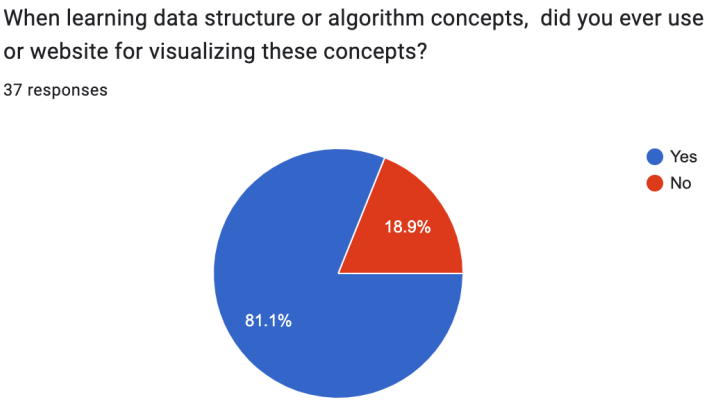


Fig. 5: Question regarding the usage of visualization apps and websites in the past.

3 Related Work

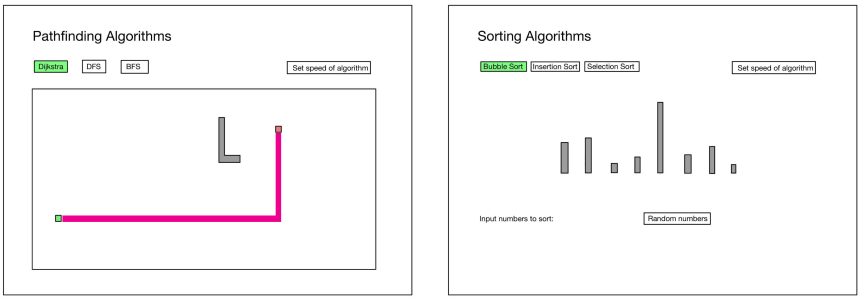
The visualization of algorithms is a crucial educational and practical tool, aiding both learners and professionals in comprehending the complex algorithms as seen in section **??**. In this section, we look into the existing algorithm visualization tools and platforms, highlighting their weaknesses and strengths. This exploration sets the context for the development of our algorithm visualization website, which aims to provide an intuitive and interactive platform for understanding various algorithms, including pathfinding and sorting algorithms.

Several algorithm visualization tools such as "Algorithm Visualizer" and "VisuAlgo" have been around for some while. These platforms offer a range of visualization techniques, making them invaluable resources for computer science students and educators. While these existing algorithm visualization tools have made significant contributions to the field, there are still opportunities for improvement. Many tools lack a comprehensive set of algorithms or may be limited in their interactivity. Furthermore, the integration of user-friendly interfaces and the incorporation of explanatory content could enhance the learning experience. We are aiming to educate users, who have little to no prior experience in the field of algorithm and want to make the learning experience as easy as possible. For example, "Algorithm Visualizer" is overwhelming for newcomers with its complexity. It is not easy to use for a quick and easy overview of how a certain algorithm work. Enhancing user-friendly interfaces and providing clear explanations can significantly enhance the learning journey for beginners. In addition to the complexity of the websites, we struggled to get the algorithm working or the website crashed, needing us to reload the website.
Our proposed algorithm visualization website builds upon the strengths of existing tool while placing user-friendliness at the forefront. By offering interactive visualization and easy-to-follow educational resources, we vision our platform as a welcoming entry point for those new to algorithms. In the subsequent sections, we will detail the key features and development for our platform.

4 Proposed Solution

Having explored other algorithm visualization tool and platforms in section 3, we now focus on our proposed solution for algorithm education. Drawing insights from the related work, we will introduce our vision for an algorithm visualization website, that seeks to redefine accessibility, engagement and educational resources in the field of algorithms. Our proposed solution is aimed at the comprehensive understanding of algorithms through interactive visualization and customizable learning experiences. Unlike static videos or limited-input-tools, our platform empowers users to actively engage with these fundamental concepts. Our website is driven by a learner-centric approach, user-friendliness and simplicity.

In this section we will go over our draft of the website. In figure 6 our drafts for the pathfinding and sorting algorithms can be seen. Figure 1 (a) shows the draft of the path finding algorithm visualizer. At the top left users can select from a range of pathfinding algorithms like Dijkstra's, DFS or BFS and visually observe how these algorithms find paths between designated start and end nodes on customizable grids. Users can add obstacles (walls) to simulate real-world scenarios. Users have the flexibility to set their own start and end nodes for pathfinding, specify the speed at which algorithms are executed. This customization allows users to explore at their own pace and adapt the learning



(a) Pathfinding (b) Sorting

Fig. 6: Draft of the website for pathfinding and sorting algorithms.

experience to their preferences. Through the option to set the speed, we will showcase the step-by-step process of the pathfinding execution by highlighting visited nodes, providing a detailed view of how algorithms operate.
In figure 1 (b) our draft for the sorting algorithms can be seen. Similarly, at the top left users are able to select different sorting algorithms like Bubble Sort, Insertion Sort and Selection Sort and visualize how these algorithms rearrange elements in a dataset represented by bars. Users are able to input their own data or use randomized inputs for sorting. The inputs will be integers. Similar to the visualization of the pathfinding algorithms, users can set the speed of the sorting algorithm to see the step-by-step process of the sorting steps.
In addition to visualization, users can access comprehensive theoretical explanations for each algorithm. These explanations are presented in a user-friendly format to ensure not only see the algorithms in action but also understand the underlying principles. To reinforce learning, we offer interactive quizzes that test user's knowledge of algorithms. These quizzes help users assess their understanding and identify areas for improvement.
As time allows, we plan to continually expand our library of algorithms and add a visualization of different data structures, providing users more resources for learning algorithms and even data structures. Our tech stack combines Node.js, React and Figma for the front end and Java Spring for the backend. We will provide an open-source JavaAPI that enables users to contribute, expand and customize the platform for their specific algorithm visualization needs. Our frontend development is centered around providing a user-friendly and interactive interface for algorithm visualization and learning. We will utilize Node.js as the runtime environment and React for building responsive and engaging user interfaces. Figma will be used for designing visually appealing and intuitive user interfaces. Our backend, powered by Java Spring, will expose endpoints for algorithm visualization and management. Users can easily extend the API to integrate additionally algorithms or customize existing ones or use the API to get results to problems, that can be solved by the API.

5 Requirement Analysis

To create a guideline for development, a requirements catalog was created using RFC 2119. This catalog encompasses the key components of a web app that is supposed to visualize algorithms in a playful manner. The catalog includes parts of the server, web page and several overarching important issues. [Bra97]

When describing the requirements, adherence to the RFC 2119 guidelines from Harvard University in Cambridge was maintained. These guidelines dictate the formulation of all requirements using specific keywords. Here are the keywords of the RFC2119:

- MUST: Required requirement to finish the development
- MUST NOT: Prohibited requirement that may not be met to finish the development
- MAY: Truly optional requirement, fulfilling this requirement can enhance the product sometimes, in other circumstances it is better to not fulfill it
- SHOULD: Recommended requirement, it is also possible to finish the project without fulfilling this requirement, fulfilling it will definitely enhance the product
- SHOULD NOT: Recommended to not fulfill requirement, fulfilling this requirement may diminish the quality of the of the product

5.1 Requirements Table

All of the requirements in the table are going to be translated to issues in Jira. Additionally, most requirements will have smaller sub-issues that will track the progress of issues from a higher level in more detail.

Table 1: Table of Requirements

Number	Description	Priority	Source	Assigned Developer
NF1	The Website must work without crashing	High	Server	All
NF2	The Layout of the website should be user friendly and not crowded	High	Frontend-Design	Matt, Jarod
NF3	Navigation must be easy and intuitiv	High	Frontend	Jarod
F3.1	Navigation through different search algorithms must be possible with a search bar	High	Frontend-Header	Jarod
F3.2	Clicking on the Algoverse Icon should link to the starting site	Low	Frontend-Header	All
NF4	Several algorithms must be included	High	Front- and Backend	All
NF5	The Algorithm must be implemented in the right way	High	Front- and Backend	All

[Som16]

6 Planning

Because of prior work experiences the work group will use a Kanban Board and the Scrum Method with the help of the project management tool Jira. The advantages of agile programming methods are flexible issues and requirements. Also developers will have an easier time tracking progress. Weekly scrum revisions will take place on sundays. All Issues on the Kanban Board will have to traverse the following steps before they are finished:

- TO-DO
- In-Development
- Bug-Fixing
- In-Revision
- Finished

All Issues for the week start in TO-DO. As soon as a devloper starts working they will take the respective issue they are working on and move them into development. As soon as they are done they can be put in In-Revision so the assigned revisor can test the feature or implementation. Moving forward there will be two paths for the issue. The first path is moving it into finished. This happens if there were no problems found with the implementation and it seems

bug free. If it needs further development because of a bug or a similar reason it is put into Bug-Fixing where it will need to be overworked till it's adapted to the notes of the revisor. After that it can be moved into In-Revision. The only way issues can move out of finished is through later tests which are conducted when different parts of the app are getting connected to each other. Should one feature break upon connecting parts of the app it will be moved into Bug-Fixing again. The following table will show the distribution of additional work between the developers. Most of these steps couldn't be made into requirements because they are too detailed.

Table 2: Table of Requirements

Week	Frontend	Backend
4	Setting up Frontend	Setting up Backenend
5	Design and CI	Implementing CI
6	Header and Starting Site	Data Structures
7	Website Structure	API for Frontend
8	Visualizing Pathfinding	Algorithm Implementation (Pathfinding)
9	Visualizing Sorting	Algorithm Implementation (Sorting)
10	Upgrade Design (if needed)	Automating Tests
11	More Algorithms	Algorithm Implementation (more)
12	More Algorithms (trees)	Algorithm Implementation (trees)
13	Tests	Tests
14	Bugfixing	Bugfixing
15	Paper	Paper
16	Presentation	Presentation

GitHub will serve as the version management tool. Branching will be handled issue wise. Every issue will get it's own branch and as soon as an issue was handled, a merge request can be created. One of the other developers can review the merge request and merge the issue into the main branch afterwards. Another feature of Git that will be utilized is the CI/CD pipeline, which will be equipped with tests to ensure the functionality of the website is always maintained.

References

[Bra97] Bradner, S. (1997). Key words for use in rfcs to indicate requirement levels. Technical Report RFC 2119, RFC Editor.

[Som16] Sommerville, I. (2016). Software Engineering.